95-702 Distributed Systems for ISM

Project 2 Client-Server Computing

Five Tasks

🏁 Submit to Canvas a ***single PDF file*** named Your_Last_Name_First_Name_Project2.pdf along with a single zip file containing each of the five IntelliJ projects below (Task 0 through 4).

The single PDF will contain your responses to the questions marked with a checkered flag. It is important that you ***clearly l*** w. It is also important to ***be prepared*** to demonstra ubmission. Be sure to provide your name and em

The five IntelliJ projects will the zip of your WHOLE IntelliJ project for tasks 0, 1 1, will contain one client and one server. Task 1 will s player in the middle. For each project, zip the whole >To Zip" in IntelliJ.

When all of your work is co files into one big zip file for submission. Name this f

Learning Objecti

Our **first objective** is for yo otocol (UDP). UDP is used in many internet applications. The Domain Name Service (DNS) and the Dynamic Host Configuration Protocol (DHCP) both use UDP. Most video and audio traffic uses UDP. Online gaming and Voice over IP (VoIP) use UDP. We use UDP when we need high performance and do not mind an occasional dropped packet.

TCP, on the other hand, is also widely used. It works hard to make sure that not a single bit of information is lost in transit. The Hyper Text Transfer Protocol (HTTP) uses TCP. We are not using TCP in this project.

Our **second objective** is to understand the implications of a malicious player in the middle.

Our **third objective** is for you to understand the abstraction provided by Remote Procedure Calls (RPC's). We do this by asking that you use a proxy design and hide communication code and keep it

separate from your application code. RPC has been used for four decades and is at the foundation of many distributed systems.

Our **fourth objective** is the learn how to distribute a stand alone application. We use a simple neural network as our application. Our intent is not to study neural networks in a class on distributed systems. But some of you might decide to dig further into neural networks and use this application as a starting point.

Optionally, you may use a large language model (based on neural networks), such as ChatGPT or Copilot, to create some of your code. Task 0, Task 1, and Task 4 must be done without the help of a large language model. There will be exam questions that ask specifically about the code in these three tasks. While you are allowed to use AI tools for these three tasks, it is totally optional. There will be questions about these tasks, too, but these questions will be more generic (since different students may code these ta

Submission note

When you are asked to sub ocumented. Points will be deducted if code is not wel ill contain a comment describing what the block c cumentation for an example of good and bad c

Rubric

See the General Course Ru shed rubric for this assignment but the general signment will be evaluated.

Some simplifications:

In all of what follows, we are concerned with designing servers to handle one client at a time. We are not exploring the important issues surrounding multiple, simultaneous visitors. If you write a multi-threaded server to handle several visitors at once, that is great but is not required. It gains no additional credit.

In addition, for all of what follows, we are assuming that the server is run before the client is run. If you want to handle the case where the client is run first, without a running server, that is great but will receive no additional credit.

In Task 1, we are assuming that the server is run before the malicious player and the malicious player is run before the client.

In this assignment, you need not be concerned with data validation. You may assume that the data entered by users is correctly formatted.

In general, if these requirements do not explicitly ask for a certain feature, then you are not required to provide that feature. No additional points are awarded for extra features.

Cite your sources

If you use any code that is not yours (including code from a large language model), you are required to clearly cite the source - include a full URL in a comment and place it just above the code that is copied. If you use a large language model to generate code, be sure to say so. Be careful to cite your sources. If you submit code that you did not create on your own and you fail to include proper citations then that will be reported as an academic violation.

Task 0 introduc J project "Project2Task0

In Task 0, you will make sev EchoClientUDP.java. Note that
these two programs are sta web application in IntelliJ.
Both of these programs wil

EchoServerUDP.java from C

```
import java.net.*;
import java.io.*;
public class EchoServerUDP{
        public static void main(String args[]){
        DatagramSocket aSocket = null;
        byte[] buffer = new byte[1000];
        try{
                aSocket = new DatagramSocket(6789);
                DatagramPacket request = new DatagramPacket(buffer, buffer.length);
                while(true){
                        aSocket.receive(request);
                        DatagramPacket reply = new DatagramPacket(request.getData(),
                request.getLength(), request.getAddress(), request.getPort());
                        String requestString = new String(request.getData());
                                                                tring);


                }
        }catch (Socket                                          ' + e.getMessage());
        }catch (IOExce                                          etMessage());
        }finally {if(a
        }
}
```

Note the difference betwee ne server uses a DatagramPacket to receive uses a DatagramPacket to send data back to the cli ways based on a byte array. So, to send a messag ne message to a byte array. To receive a message from a DatagramPacket, we must convert the byte array to a String message (if we are expecting a String message).

Note below how the client does the same thing. The client wants to send a String message. So, it extracts a byte array from the String (the variable m). And we then use m to build the DatagramPacket.

When the client receives a reply, the method reply.getData() returns a byte array - which we use to build a String object.

EchoClientUDP.java from Coulouris text

```
import java.net.*;
import java.io.*;
public class EchoClientUDP{
    public static void main(String args[]){
        // args give message contents and server hostname
        DatagramSocket aSocket = null;
        try {
                InetAddress aHost = InetAddress.getByName(args[0]);
                int serverPort = 6789;
                aSocket = new DatagramSocket();
                String nextLine;
                BufferedReader typed = new BufferedReader(new InputStreamReader(System.
                while ((nextLine = typed.readLine()) != null) {
                  byte [] m = nextLine.getBytes();
                  Data                                          m.length, aHost, serv
                aSocke
                byte[]
                Datagr                                          , buffer.length);
                aSocke
                System                                          String(reply.getData()))
    }

        }catch (Socket                                          Exception: " + e.getMess
        }catch (IOExce                                          " + e.getMessage());
        }finally {if(a
    }
}
```

0. Get these programs running in Intellij. The two programs are placed in the same Intellij project and you are provided with two windows to interact with the two programs. Make the following modifications to the client and the server.
1. Change the client's "arg[0]" to a hardcoded "localhost".
2. Document the client and the server. Describe what each line of code does.
3. Add a line at the top of the client so that it announces, by printing a message on the console, "The UDP client is running." at start up.
4. After the announcement that the client is running, have the client prompt the user for the server side port number. It will then use that port number to contact the server. For now, enter 6789.
5. Add a line at the top of the server so that it announces "The UDP server is running." at start up.
6. After the announcement that the server is running, have the server prompt the user for the port number that the server is supposed to listen on. Enter 6789 when prompted.